Software Systems Verification and Validation

Lecture 11b: Model checkin

Spin Model Checker

Assoc. Prof. Andreea Vescan

Babeș-Bolyai University

Cluj-Napoca

2019-2020

Outline

- Spin
- Promela Model
  - Statements
  - Examples
- Concurrency and Interleaving Semantics
  - Examples
- Linear Temporal Logic
  - Examples
- JSpin

- Questions

Model checking

Spin

- Developed at Bell Labs.
- In 2002, recognized by the ACM with Software System Award.
- SPIN (= Simple Promela Interpreter)
- is a tool for analyzing the logical consistency of concurrent systems
- Concurrent systems are described in the modelling language called Promela (= Protocol/Process Meta Language)

Promela

- Promela (= Protocol/Process Meta Language)
- allows for the dynamic creation of concurrent processes.
- communication via message channels can be defined to be
  - synchronous (i.e. rendezvous),
  - asynchronous (i.e. buffered).

Promela Model

- Promela model consist of:
  - type declarations
  - channel declarations
  - variable declarations
  - process declarations
  - [**init process]**
- A process type (**proctype) consist of**
  - a name
  - a list of formal parameters
  - local variable declarations
  - Body

- A process
  - is defined by a **proctype definition**
  - executes concurrently with all other processes, independent of speed of behaviour
  - communicate with other processes
    - using global (shared) variables
    - using channels
- There may be several processes of the same type.
- Each process has its own local state:
  - process counter (location within the **proctype)**
  - contents of the local variables

Statements

- The body of a process consists of a sequence of statements.
- A statement is either
  - executable: the statement can be executed immediately.
  - blocked: the statement cannot be executed.
- An assignment is always executable.
- An expression is also a statement; it is executable if it evaluates to non-zero
- The **skip statement is always executable.**
  - "does nothing", only changes process' process counter
- A **printf statement is always executable (but is not** evaluated during verification, of course).
- **assert(<expr>);**
  - The **assert-statement is always executable.**
  - If **<expr> evaluates to zero, SPIN will exit with an error, as**
- the **<expr> "has been violated".**
  - The **assert-statement is often used within Promela models,**
- to check whether certain properties are valid in a state.

Examples (01 Simple Examples)

- ReversingDigits.pml
  - Check
  - Random
- DiscriminantOfQuadraticEquation.pml
  - Check
  - Random
- NumberDaysInMonth.pml
  - Check
  - Random
- MaximumNondeterminism.pml
  - Check
  - Random
  - “Branch 1” and “Branch 2”
- Maximum –second example-MaximumIfElse.pml
  - Check
  - Random
- GCD.pml
  - Check
  - Random
- IntegerDivison01.pml
  - Check
  - Random

Concurrency and Interleaving Semantics

02 Concurrency and interleaving semantics

- Promela processes execute concurrently.
  - Non-deterministic scheduling of the processes.
  - Processes are interleaved (statements of different processes do not occur at the same time).
  - exception: rendez-vous communication.
- All statements are atomic; each statement is executed without interleaving with other processes.
- Each process may have several different possible actions enabled at each point of execution - only one choice is made, non-deterministically.
- InterleavingStatements.pml
  - Check
  - Random
  - 6 possibilities of the execution
    - n1,p,n2,q;
    - n1,n2,p,q;
    - n1,n2,q,p;
    - n2,q,n1,p;
    - n2,n1,q,p;
    - n2,n1,p,q.
  - Interactive simulation – Interactive button
- InterferenceBetweenProcesses.pml
- InterferenceBetweenProcessesDeterministic.pml

Examples 03 Critical section

- CriticalSection_Incorrect.pml
  - both processes – in the critical section
- CriticalSection_MutualExclusion.pml – not satisfied
  - Mutual exclusion – at most one process is executing its critical section at any time.
- CriticalSection_With_Deadlock.pml
  - Blocking on an expression – user Interactive simulation
  - Absence of deadlock – it is impossible to reach a state in which come processes are trying to enter their critical sections, but no process is successful.
- CriticalSection_SolutionAtomic.pml
  - The atomic sequence may be blocked from executing, but once it starts executing, both statements are executed without interference from the other process.

Linear Temporal Logic

- Temporal logic formulae can specify both safety and liveness properties.

- LTL ≡ propositional logic + temporal operators

`[]P`	always P
`<>P`	eventually P
`P U Q`	P is true until Q becomes true

Examples 04 LTL examples

- CriticalSection_MutualExclusionLTL.pml
  - LTL formula:
    - []mutex
  - Translate
  - Verify
- CriticalSection_MutualExclusionLTL02.pml
  - LTL formula:
    - []mutex
  - Translate
  - Verify
- CriticalSection_With_Starvation.pml
  - LTL formula:
    - <>csp
  - Translate
  - Acceptance
  - Verify

JSpin

- http://spinroot.com/
- Installation JSpin

  http://jspin.software.informer.com/5.0/

Questions

- Thank You For Your Attention!

References

Sources

[1] Baier Christel, Katoen Joost-Pieter, Principles of Model Checking , ISBN 9780262026499, The MIT Press, 2008

- Chapter 1 - System verification, Chapter 2 – Modelling Concurrent systems (pag. 19-20), Chapter 3 (pag. 89, 107, 120-121), Chapter 5 – Linear Temporal Logic (pag. 229-233), Chapter 6 – Computation Tree Logic (pag. 313-323)

[2] Ben-Ari, Mordechai, Principles of the Spin Model Checker, ISBN 978-1-84628-770-1, Springer-Verlag London, 2008